Real Time Direct Visual Odometry for Flexible Multi-Camera Rigs

Benjamin Resch, Jian Wei and Hendrik P.A. Lensch

Computer Graphics Group, University of Tübingen

Abstract. We present a Direct Visual Odometry (VO) algorithm for multi-camera rigs, that allows for flexible connections between cameras and runs in real-time at high frame rate on GPU for stereo setups. In contrast to feature-based VO methods, Direct VO aligns images directly to depth-enhanced previous images based on the photoconsistency of all high-contrast pixels. By using a multi-camera setup we can introduce an absolute scale into our reconstruction. Multiple views also allow us to obtain depth from multiple disparity sources: static disparity between the different cameras of the rig and temporal disparity by exploiting rig motion. We propose a joint optimization of the rig poses and the camera poses within the rig which enables working with flexible rigs. We show that sub-pixel rigidity is difficult to manufacture for $\geq$720p cameras which makes this feature important, particularly in current and future (semi-)autonomous cars or drones. Consequently, we evaluate our approach on own, real-world and synthetic datasets that exhibit flexibility in the rig beside sequences from established KITTI dataset.

1 Introduction

Visual Odometry (VO) and Self Location And Mapping (SLAM) systems traditionally are feature-based approaches: Distinct features are tracked over many frames. The scene reconstruction, i.e. determining for all frames the parameters of the camera poses and the positions of all scene points, is usually formulated as a least squares optimization problem that aims at minimizing the difference between the observed feature positions in the images and the reprojections of the scene points into the frames.

Direct methods instead maintain per pixel depth information for several keyframes which allows the keyframe's image to be projected into other frames, given the camera's intrinsic parameters and its relative pose to the keyframe. The camera poses of new frames can be found by minimizing the photometric error between the keyframe projected into the new frame and the new frame itself. When the relative camera transformation between the keyframe and another frame is known, depth information of the keyframes is improved with stereo depth estimation techniques.

With a stereo camera rig, the quality of the depth maps of the keyframes can be improved by applying not only the *dynamic* stereo between a keyframe and its subsequent frames, but also by applying *static* stereo between two images

that were recorded by the rig at the same time with known rig intrinsics (poses of the cameras relative to the rig) [1].

It turns out that direct VO methods are much less forgiving to bad calibrations as feature-based VO methods. Feature-based methods optimize based on distances on the image which increase gradually when the calibration becomes bad. Direct methods optimize based on photoconsistency in a small window which can be arbitrarily bad if the calibration is just one pixel off.

While the intrinsic parameters of cameras can usually be controlled with pixel precision and the tracking of the rig's pose works just as precise, the rig intrinsics can't be assumed to be static enough for sub pixel precision over time, especially for higher resolution cameras, large baselines and/or material/cost/weight limitations of the rig. Deflection theory shows that two 0.5 kg cameras mounted on a 300x25x4 mm stainless steel carrier are rotated by 0.56° against each other when an acceleration of ±1 g is applied (see supplemental material). This already introduces significant errors if the rig flexibility is not concerned (see Fig. 1).

In this paper, we propose a method that extends existing, direct monocular or stereo VO approaches with per frame rig intrinsics tracking for multi-camera rigs. We target on high-frequency rotations of the cameras in the rig up to a few degrees which are hard to avoid but harmful to conventional direct VO methods (see Fig. 1,2). For each keyframe, we maintain per pixel depth information for the images of all cameras. For every new frame, we optimize for the rig's pose as well as for its intrinsics based on the photometric error between every image of the keyframe and every new image. This way, we find consistent poses for all cameras of the rig by utilizing the information that is available in all images.

In fact, we can even choose (1) the pairs of keyframe / new frame images that should be used for tracking and (2) the new frames' or keyframes' images that should be used to improve the depth information of the keyframe images. This flexibility can be used to balance between computation speed and reconstruction quality as well as to adjust the processing to the rig configuration, e.g. avoid direct interaction of cameras that never see the same field of view (FoV).

We implemented most of our algorithm on the GPU using CUDA to address the increased computational demands compared to a monocular setup.

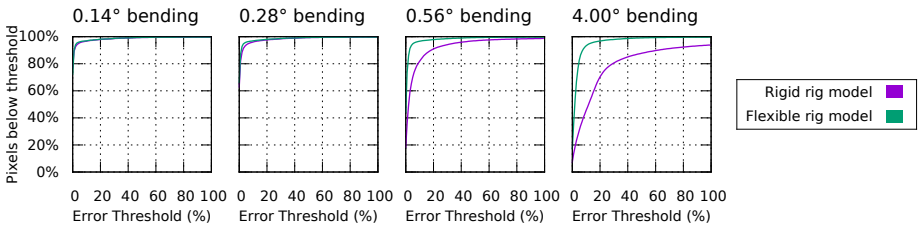

Fig. 1. ROC evaluation of the depth quality on a synthetic dataset (720p, 90° FoV camera). Note that even for small camera rotations inside the rig of 0.56°, taking the rig flexibility into account improves the results very significantly: For a 3% threshold, the recall is 49.4% (rigid model) vs. 85.9% (flexible model).

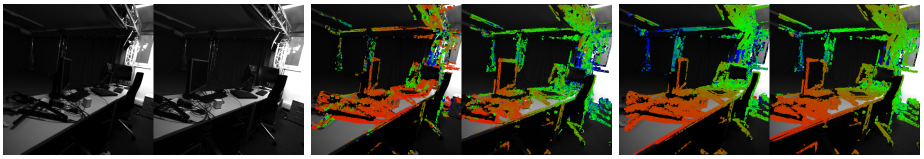

Fig. 2. Comparison of the depth maps reconstructed without (center) and with (right) flexible rig optimization. The original frames are shown on the left. Without flexible tracking, tracked frames cannot be aligned consistently to the last keyframe. This leads to a bad alignment of projected depths onto image gradients and therefore to stereo errors which result in incomplete and noisy depth maps (center), while flexible tracking does not suffer from those issues (right).

Please note that building a globally consistent reconstruction of a scene including loop closing and global error distribution is out of the scope of this document. However, our technique can be used as a plugin replacement for the VO part of other SLAM systems.

2 Related Work

The proposed approach reconstructs camera poses as well as the scene structure, so we consider VO / SLAM methods as well as Structure from Motion (SfM) and Multi View Stereo (MVS) techniques in this section.

2.1 Visual Odometry and SLAM

Most VO and SLAM methods are feature-based. Chiuso et al. [2] presented one of the first real time capable, monocular, feature-based SLAM systems. Nistér et al. [3] established the term "Visual Odometry" by presenting a feature tracking based system for frame-to-frame monocular or stereo camera pose estimation. Davison et al. [4] proposed MonoSLAM which is an Extended Kalman Filter (EKF) based method that creates a probabilistic but persistent map of natural landmarks and is therefore drift free in small workspaces. PTAM [5] is designed to avoid the iterative tracking and mapping per frame by separation into two concurrent threads: The mapping thread integrates all previously tracked camera poses and features into a consistent representation in the background while the tracking thread tracks every new frame against the newest available map. Paz et al. [6] combine an EKF-based framework with a stereo camera setup.

Direct methods for VO and SLAM are available for some years now and have two major benefits over the feature-based methods: First, there is no need to craft a feature detector. Second, not only the information from sparse feature points but virtually any gradient information in the images can be used for reconstruction. Direct monocular methods first appeared in the RGBD domain [7, 8] where the per pixel depth information comes directly from the sensor and only the tracking has to be performed. The first direct real-time method relying

solely on color images is LSD-SLAM [9] which tracks each new frame to the previous, depth-enhanced keyframe based on dense image alignment. Then, the keyframe's depth information is updated by probabilistically merging it with the depth information obtained from stereo with the previously tracked frame. LSD-SLAM was extended to stereo cameras [1] where tracking only happens on the left camera. This is in contrast to our approach which allows for tracking between multiple cameras. Consequently, in [1], a depth map is maintained only for the left camera; the right images are just used for static stereo with the corresponding left images which improves the keyframe's left depth map and introduces an absolute scale to the reconstruction. This is however not feasible if the rig intrinsics change over time. In addition, we show that we can reconstruct datasets from rigs with few overlap between the fields of view of the cameras by utilizing the information from all cameras where methods like [1] fail (see Section 4.4). Pillai et al. [10] accelerates the depth estimation for stereo pairs by starting from few sparse, Delaunay triangulated piecewise planar surfaces which can be refined in multiple iterations in areas where the matching cost is high, e.g. due to non-planarity. A substantially different approach is the method of Comport et al. [11] which avoids the explicit reconstruction of scene depth and uses the quadrifocal constraints from two pairs of stereo images to obtain the transformation of the camera rig instead.

2.2 Structure from Motion and Multi View Stereo

SfM (usually feature based) with subsequent MVS (dense, direct surface reconstruction) is similar to SLAM since both reconstruct consistent 3D models and camera poses from images. However, SfM+MVS methods traditionally aim more at image collections instead of videos and reconstruction quality instead of real time processing with low latency. Recently, there have been efforts to develop SfM / MVS methods that utilize the redundancy of video streams to achieve close-to-real-time performance. Resch et al. [12] have accelerated SfM by the consequent subsampling of frames, features, scene points and loop closure candidates in their Bundle Adjustment framework and employing a linear solver to keep unsampled frames consistent. The MVS methods in [13, 14] compute depth maps based on the photoconsistency with up to 100 other images but compare for photoconsistency based on 1x1 pixel masks only.

Delaunoy and Pollefeys propose a Photometric Bundle Adjustment [15] which is designed to recover camera parameters and a dense surface mesh based on the photometric error between observed and model-based, generated images.

3 Multi-view VO for flexible camera rigs

In this section we'd like to introduce our method. Fig. 3 shows an overview of the originally monocular LSD-SLAM algorithm [9] and our extensions.

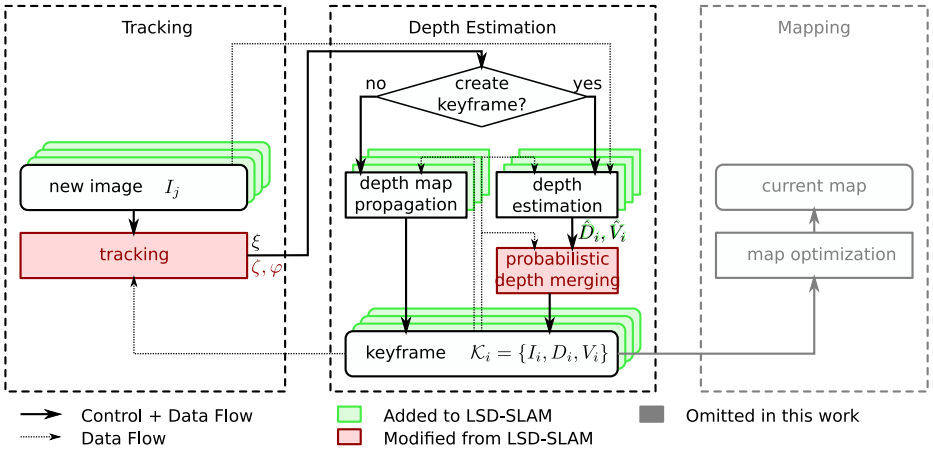

Fig. 3. Overview over the LSD-SLAM algorithm and our extensions.

LSD-SLAM tracks the camera poses of new images against a previous, depth enhanced keyframe. If the current keyframe is still good for tracking (content similar to the new frame), new depth information is estimated with a stereo method between the new image and the keyframe and is merged into the keyframe. If the keyframe is not good enough anymore, it gets propagated to the current image. A mapping component finally cares about global consistency of all keyframes.

Our method allows for multiple input images from multiple cameras per frame and stores depths for all cameras of a keyframe, consequently. We extended the tracking to optimize for the rig pose and the rig intrinsics (camera poses within the rig) jointly (Section 3.2). For stereo cameras, we propose a reliable rig parameterization in Section 3.3. Tracking gives us the transformation from every keyframe image to every new frame's image, so we can generate multiple stereo observations per keyframe (Section 3.4). In order to utilize all that information, we extended the Bayesian depth merging approach (Section 3.5).

However, we'd like to start by introducing the aspects of LSD-SLAM [16, 9] that are important to our method first:

3.1 LSD-SLAM

LSD-SLAM is a direct, semi-dense, monocular Self Location And Mapping approach. It consists of the following steps (see Fig. 3):

1. Tracking has the goal of finding the relative transformation ξ_{ij} of the camera from the last keyframe $\mathcal{K}_i = \{I_i, D_i, V_i\}$ to the most current image I_j. For keyframes, a semi-dense, inverse depth map D_i as well as the variances V_i of the inverse depth values are assumed to be available beside the pixel intensities I_i.

A new image is aligned to the previous keyframe by Levenberg-Marquard (LM) minimization of the photometric error

$$E(\xi_{ij}) = \sum_p (I_i(p) - I_j(\omega(p, D_i(p), \xi_{ij})))^2 \tag{1}$$

between every pixel p in the keyframe's image I_i and the intensity at the corresponding location in the new image I_j, obtained by warping p based on its depth $D_i(p)$ and the camera transformation ξ_{ij}. The intrinsic camera calibration $\pi : \mathbb{R}^3 \longrightarrow \mathbb{R}^2 \times \mathbb{R}$ transforming from view space to an image position plus depth is used to define the warping function

$$\omega(p, d, \xi) = [(\pi \circ \xi^{-1} \circ \pi^{-1})(p, d)]_p. \tag{2}$$

Note that ξ is the transformation of the camera, so ξ^{-1} has to be used to transform the points relative to the camera.

Tracking is performed on an image pyramid in a coarse-to-fine manner: A coarse representation helps to converge even with high disparities, fine details provide exact tracking at the end.

After tracking, a decision about creating a new keyframe is made based upon the distance that the camera covered since the last keyframe and the number of pixels that could be used for tracking. Depending on the decision, steps 2 and 3 or step 4 are executed.

2. Depth estimation produces a new inverse depth map $\hat{D}_i$ and a corresponding variance map $\hat{V}_i$, using a stereo method on the image intensities I_i and I_j by using the previously obtained camera transformation ξ_{ij}.

Depth $\hat{D}_i$ is obtained for all pixels that exhibit a sufficiently large gradient $\nabla I_i(p)$ along the Epipolar Line (EPL) by brute force evaluation of the Sum of Squared Differences (SSD) error in a 5x1 pixel mask (aligned with the EPL). The depth is further refined by second order Taylor approximation. The search range for each pixel p on the EPL corresponds to the depth range $(D_i(p) \pm 3V_i(p))$.

The variance values $\hat{V}_i(p)$ are estimated based on the photometric disparity error ($\approx \frac{\textit{camera noise}}{|\nabla I_i(p)|}$) and the geometric disparity error ($\approx 1/\langle \frac{\nabla I_i(p)}{|\nabla I_i(p)|}, \frac{EPL}{|EPL|} \rangle$, i.e. it is large if the image gradient is orthogonal to the EPL direction), both determined in the image domain and transformed to the inverse depth domain.

3. Probabilistic depth merging is used to update the previous keyframe depths D_i and variances V_i with the new estimations $\hat{D}_i$ and $\hat{V}_i$. This is done by multiplying the distributions according to the update step in a Kalman filter: Given a prior distribution $\mathcal{N}_{prior} = \mathcal{N}(D_i(p), V_i(p))$ and a new observation $\mathcal{N}_{new} = \mathcal{N}(\hat{D}_i(p), \hat{V}_i(p))$, the posterior is given by

$$\mathcal{N}_{post} = \mathcal{N}_{prior} \cdot \mathcal{N}_{new} = \mathcal{N}\left(\frac{V_i(p)\hat{D}_i(p) + \hat{V}_i(p)D_i(p)}{V_i(p) + \hat{V}_i(p)}, \frac{V_i(p)\hat{V}_i(p)}{V_i(p) + \hat{V}_i(p)} \right). \tag{3}$$

After depth and variance have been updated, smoothing and hole filling are applied to D_i and V_i.

4. Depth map propagation establishes a new keyframe by propagating depth and variance information from the previous keyframe to the currently processed frame. A forward mapping from the keyframe to the new frame is established by using ω (see Equation 2) and depths and variances are assigned to the closest pixel in the new frame's image.

After the depth propagation, the new depth and variance maps are subject to removal of occluded pixels, hole filling and smoothing.

5. Map Optimization Every keyframe is added to the mapping component of LSD-SLAM which integrates it into a complete, globally consistent map of the reconstructed scene, caring about loop closing and tracking error distribution over the whole scene. This fifth step is out of scope of our method and just mentioned for completeness.

3.2 Tracking flexible multi-camera rigs

The following subsections describe our contributions for enabling multi-camera, flexible rig VO, starting with the tracking part where we optimize for the rig pose and the rig intrinsics concurrently:

For multi-camera rigs, if the ith frame is a keyframe $\mathcal{K}_i^l = \{I_i^l, D_i^l, V_i^l\}$, it contains image intensities I, inverse depths D and inverse depth's variances V for every camera l of the rig. For tracking, we introduce the extrinsic rig transformation ζ_{ij} and the rig intrinsics $\varphi_i = (\varphi_i^{c_1}, \varphi_i^{c_2}, ...)$ that can be different for every frame i and expand to a rig-to-view-space transformation for every individual camera of the rig.

For using potentially all information that is available in all images of the keyframe and the frame to be tracked, the photometric error expands to

$$E(\zeta_{ij}, \varphi_j) = \sum_{c_i} \sum_{c_j} \left(f_t(c_i, c_j) \sum_p (I_i^{c_i}(p) - I_j^{c_j}(\omega(p, D_i(p), \zeta_{ij}, \varphi_i^{c_i}, \varphi_j^{c_j})))^2 \right). \tag{4}$$

c_i and c_j are iterating over the individual cameras of the keyframe's or the tracked frame's rig. The mapping $f_t : \mathbb{N}^2 \to \{0, 1\}$ is user defined and can be used to disable pairs of cameras for tracking, e.g. if their fields of view do not overlap at all or for performance reasons (see also Fig. 6).

For the extended photometric error, we need an extended warping function that projects an image position p and its depth d to the viewspace of a keyframe's camera, then to the keyframe's rig space, then to the tracked frame's rig space, to the tracked camera's view space and into the tracked camera's image:

$$\omega(p, d, \zeta, \varphi_i^{c_i}, \varphi_j^{c_j}) = [(\pi \circ \varphi_j^{c_j} \circ \zeta^{-1} \circ \varphi_i^{c_i\,-1} \circ \pi^{-1})(p, d)]_p. \tag{5}$$

Note that for introducing a new rig configuration to our system, it is sufficient to specify φ_i, i.e. a rig-to-view-space transformation for each camera (for flexible rigs: based on some rig parameters which may change for each frame i).

For solving the minimization problem with the LM algorithm, we have to solve the augmented normal equations (please refer to [17] for its derivation)

$$(J^T J + \mu I)\delta_{\zeta\varphi} = J^T \epsilon \tag{6}$$

for $\delta_{\zeta\varphi} = (\delta_\zeta, \delta_\varphi)$ to improve our parameters with $\zeta_{n+1} = \zeta_n + \delta_\zeta$ and $\varphi_{n+1} = \varphi_n + \varphi_\zeta$. μ is a damping factor which controls if the updates $\delta_{\zeta\varphi}$ are Gauss Newton or small gradient descent steps. $J = (\frac{\partial E}{\partial \zeta_1}, ..., \frac{\partial E}{\partial \zeta_6}, \frac{\partial E}{\partial \varphi_1}, \frac{\partial E}{\partial \varphi_2}, ...)$ is the Jacobian matrix, containing the derivatives of the error function to all rig pose and rig intrinsics parameters. Note that each $\frac{\partial E}{\partial ...}$ is a column vector with one element for each $()^2$ term of Equation 4. We know that the pose of one camera on the rig just has 6 degrees of freedom (DoF), so we prefer to evaluate $J_c = (\frac{\partial E}{\partial \xi_1}, ..., \frac{\partial E}{\partial \xi_6})$. To achieve this, we approximate the known transformation from rig pose ζ and rig intrinsics φ to camera pose ξ locally and linearly for every camera c with the matrix

$$J_c^{\zeta\varphi\to\xi} = (R|C_c) \qquad R = \left(\frac{\partial \xi_i}{\partial \zeta_j}\right)_{i,j=1...6} \qquad C_c = \left(\frac{\partial \xi_i}{\partial \varphi_j^c}\right)_{i=1...6;j=1...} . \tag{7}$$

By using this transformation, we can calculate $\delta_\xi = J_c^{\zeta\varphi\to\xi}\delta_{\zeta\varphi}$. From the chain rule it follows that we can replace $J = J_c(J_c^{\zeta\varphi\to\xi})^T$ in Equation 6.

This way, we only have to evaluate the minimum number of derivatives but are able to transform them to a Jacobian for any arbitrary rig parameterization. In addition, this helps to keep the GPU code independent from the rig parameterization and allows for implementing new rigs with different camera configurations easily at one place.

3.3 A flexible stereo rig model

Generally, a stereo rig model has 6 DoF: Trivially one would use an identity camera-to-rig transformation for one of the cameras in the rig while the other one can be placed freely in the rig space. To avoid scale drifting, one might want to set the distance between the cameras fixed which leaves 5 DoF remaining. We found that this is a valid assumption even for flexible camera rigs because bending a rig by small angles leads to negligible changes in the distance between the cameras (e.g. 0.5° bending $\Rightarrow 10^{-5}\%$ closer).

In most stereo camera rigs, both cameras are attached to the rig in the same way and the rig is mounted or hold symmetrically, so we prefer a symmetrical rig model as well to have a more natural mapping from real world effects to parameter space.

For our first experiments we used the rig model shown in Fig. 4 on the left. It exhibits individual parameters for the cameras' roll and pan, but a common tilt parameter. Using individual tilt parameters would introduce 6 DoF which is above the minimum and introduces an ambiguity since tilting the rig can also be accomplished by tilting both cameras.

We observed that this rig model tends to shear as shown in Fig. 5 when the scene scale starts to drift instead of fixing the scale. Therefore we switched

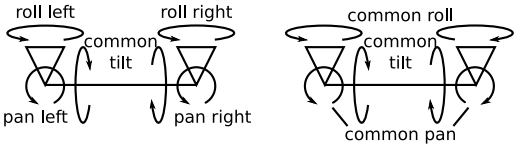

Fig. 4. Comparison of symmetric stereo rig models. The left model exhibits the theoretical minimum number of DoFs for fixed baselines. The right model is the one that we finally used due to its clearly distinct DoFs.

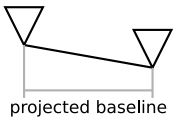

Fig. 5. Degeneration of the 5 DoF rig model. Cameras may shift in front of each other to reduce the projected baseline of the rig, overriding the scale fixation that it should actually introduce.

to the 3 DoF model shown in Fig. 4 on the right which ensures that the rig's baseline stays orthogonal to the average camera's viewing direction. Although this model is not able to represent asymmetric camera rotations correctly, it provides superior reconstruction quality on exactly such input data.

Depth / panning ambiguity While using the 3 DoF stereo rig model, we observed that the depth tends to get compressed in front of the cameras while the cameras start panning towards each other after a few hundred frames. Although depth and panning are not really ambiguous (i.e. the photometric error $E(\zeta_{ij}, \varphi_j)$ is always smaller with the correct panning), it seems like the constraints in the images are not strong enough to discern them clearly.

To resolve this issue, we implemented a low frequency panning correction: We assume that the camera rotations relative to the rig are high frequent and on average corresponding to the initial rig parameterization. Therefore, we keep track of the low pass filtered panning pan_{avg} and correct each optimized panning parameter for the difference between the initial and the averaged panning:

$$pan_{avg}^{n+1} = (1-\lambda) pan_{avg}^{n} + \lambda pan^{n+1} \tag{8}$$

$$pan_{correction} = pan_{avg}^{n+1} - pan_{init} \tag{9}$$

$$pan_{corrected}^{n+1} = pan^{n+1} - pan_{correction} \tag{10}$$

This ensures that the panning parameter is unable to drift away for many subsequent frames. We got good results on all of our datasets with $\lambda = 0.1$.

3.4 Depth estimation

We use the depth estimation component of LSD-SLAM in Section 3.1 as it is but apply it potentially once for each possible pair of a keyframe camera and the tracked frame's camera. We can simply obtain the camera-to-camera transformation

$$\xi_{ij}^{c_k c_l}(x) = \varphi_j^{c_l}(\zeta_i j(\varphi_i^{c_k\,-1}(x))) \qquad |x \in \mathbb{R}^3 \tag{11}$$

that is necessary for stereo depth reconstruction by concatenating the rig intrinsics and rig transformations obtained during tracking.

3.5 Probabilistic depth merging with multiple observations

In this subsection we examine how to merge multiple depth observations that come from stereo with the different images from the tracked frame into the depth and variance maps of *one* of the cameras c of keyframe. Let's consider one pixel p of the keyframe whose depth is described as a normal distribution $\mathcal{N}_{prior} = \mathcal{N}(D_i^c(p), V_i^c(p))$ and the depth distributions $\mathcal{N}_{c_j \in C} = \mathcal{N}(\hat{D}_i^{cc_j}(p), \hat{V}_i^{cc_j}(p))$ of the corresponding pixels in the stereo observations with the tracked frame's cameras c_j. To perform a Bayes filter update step, we need the joined distribution $\mathcal{D}_{new}$ of the observations which can be obtained by

$$\mathcal{D}_{new} = \sum_{c_j \in C} \mathcal{N}_{c_j}. \tag{12}$$

The update step can then be expressed by

$$\mathcal{D}_{post} = \mathcal{N}_{prior} \cdot \mathcal{D}_{new} = \mathcal{N}_{prior} \cdot \sum_{c_j \in C} \mathcal{N}_{c_j}. \tag{13}$$

Now $\mathcal{D}_{new}$ and $\mathcal{D}_{post}$ are not normal distributions but we will have to represent them as one to update the keyframe's maps. Therefore we rewrite our formulation to apply the multiplications of the distributions first (the product of two normal distributions is again a normal distribution) and approximate the sum by one single normal distribution, i.e. we do the approximation of our distribution as normal distribution as late as possible:

$$\mathcal{N}_{post} \approx \sum_{c_j \in C} \left(\mathcal{N}_{prior} \cdot \mathcal{N}_{c_j}\right) \tag{14}$$

To calculate $\mathcal{N}_{post}$, we have to extend our model of normal distributions by a weight parameter: $\mathcal{N}(d, v, w)$. It was previously omitted because we were dealing with probability distributions whose weights and integrals evaluate to one always. In the sum above, however, we need this weight since we want its terms to contribute differently to the resulting distribution, depending on how much $\mathcal{N}_{prior}$ and $\mathcal{N}_{c_j}$ overlap.

Formulas for computing the product or the merge of normal distributions can be found in [18]:

Product of normal distributions: $\mathcal{N}(d, v, w) = \mathcal{N}(d_1, v_1, w_1) \cdot \mathcal{N}(d_2, v_2, w_2)$

$$d = \frac{d_1 v_2 + d_2 v_1}{v_1 + v_2} \qquad v = \frac{v_1 v_2}{v_1 + v_2} \qquad w = \frac{\mathcal{N}(d; d_1, v_1, w_1) \cdot \mathcal{N}(d; d_2, v_2, w_2)}{\mathcal{N}(d; d, v, 1)} \tag{15}$$

Merge of normal distributions: $\mathcal{N}(d, v, w) \approx \sum_i \mathcal{N}(d_i, v_i, w_i)$

$$w = \sum_i w_i \qquad d = \frac{1}{w} \sum_i w_i d_i \qquad v = \sum_i \frac{w_i}{w} (v_i + (d_i - d)^2) \tag{16}$$

Note that our implementation contains a $f_{m_d}(c_i, c_j) : \mathbb{N}^2 \to \{0, 1\}$ similar to f_t from Section 3.2 to allow the user to control which tracked frame cameras c_j are used to update the the maps of keyframe camera c_i (see also Fig. 6).

3.6 Depth map propagation

We propagate depths and variances from one keyframe to the next within the same camera. When a new keyframe is created, we potentially apply one iteration of static stereo, i.e. update each camera's depth and variance maps with the stereo observations from the other cameras of the same frame. This can be user-controlled with $f_{m_s}(c_i, c_j) : \mathbb{N}^2 \rightarrow \{0, 1\}$.

4 Evaluation

We evaluated different configurations (see Fig. 6) of our algorithm on various datasets. Please note that this is a frame-to-frame VO algorithm that does not perform full SLAM including loop closing and global optimization and is therefore not directly comparable to such methods.

4.1 KITTI dataset

Results for two scenes of the KITTI odometry benchmark [19] based on a stereo setup are shown in Fig. 7 using the configurations displayed in Fig. 6.

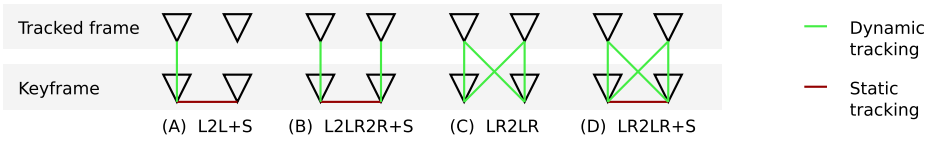

Fig. 6. Different camera tracking and mapping configurations. Dynamic tracking is either done within a single stream, within all streams or across streams. Static tracking is performed between two images captured at the same time.

Dynamic tracking and mapping on the left camera only is analog to [1] and produces similar results. Worse results with other configurations of our algorithm, e.g. LR2LR+S, have also been observed by [1] and seem to be related to more outliers due to obstructions for large baselines between different cameras. However, those other configurations are crucial for flexible camera rigs (Section 4.2) or rigs with barely overlapping FoVs (Section 4.4).

Please refer to the supplemental material for four camera KITTI evaluations.

4.2 Flexible rigs

We evaluate our estimation of intrinsic rig parameters based on synthetic and real-world data. Fig. 8 compares the intrinsic rig parameters to the ground truth of a synthetic scene (highly textured, 720p, with symmetric rig intrinsics as in Section 3.3) and shows that our approach clearly produces correct results.

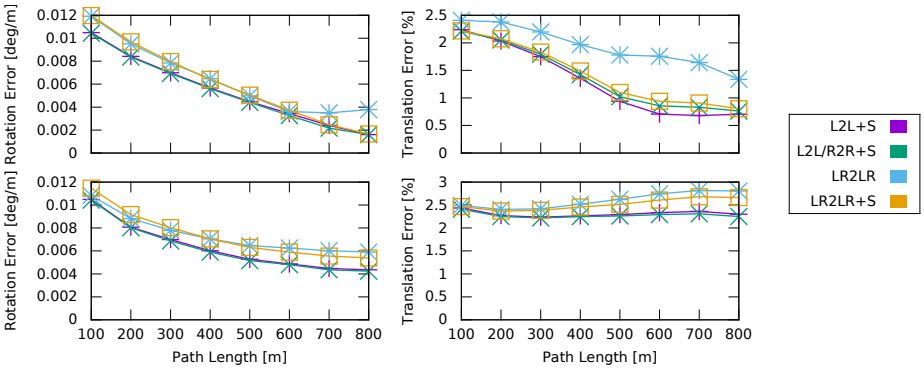

Fig. 7. Results from the KITTI odometry benchmark scenes 06 (top) and 08 (bottom). Note that static stereo seems to improve quality significantly. Tracking and depth estimation between different cameras from different frames produces worse results in the scenes because of errors that are introduced by more occlusions on longer baselines.

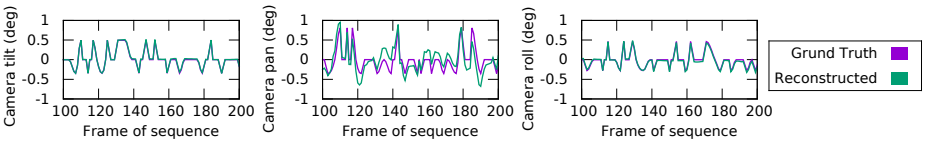

Fig. 8. Evaluation of the rig intrinsics reconstruction on synthetic data with ground truth. Note that roll and tilt are reconstructed nicely while the panning exhibits some artifacts due to our correction (Section 3.3) which we accept in favor of drifts or a collapsing reconstruction.

For evaluation on real world data, we recorded three scenes with a flexible stereo camera rig with a FoV of 110 degrees and 2048x2048 pixels per image downsampled to 512x512 pixel images with 30 fps. The rig was bended and twisted by about 3 degrees while recording. Fig. 9 compares the drift of the camera pose when sequences are reconstructed based on a rigid or a flexible rig model. To evaluate the drift, we moved the camera back to its origin at the end of the sequence and compare the first and last reconstructed rig pose. Reconstructed depth maps for stereo pairs are shown in Fig. 2. Our results show that dense algorithms that don't keep track of the rig intrinsics perform significantly worse and introduce jittering to the rig pose since the tracking cannot converge on all cameras consistently. Reconstructions for this section were performed using LR2LR+S (see Fig. 6), because we observed unstable reconstructions without cross-camera tracking on some of our flexible rig scenes.

4.3 Timings

Fig. 10 compares the per frame processing times for different resolutions, different configurations and different steps of the algorithm. It shows that there is just

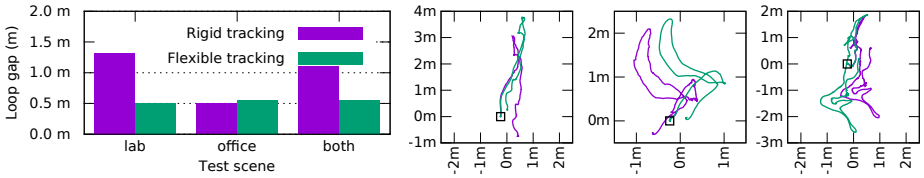

Fig. 9. Loop gap evaluation on a flexible rig dataset. Tracking with a flexible rig model clearly improves the drift in the tracking. A closer look to the plotted paths reveals that tracking a flexible rig with a rigid model leads to jittering in the reconstructed camera positions, most likely caused by convergence to one of the cameras when the cameras don't agree due to a deformed rig.

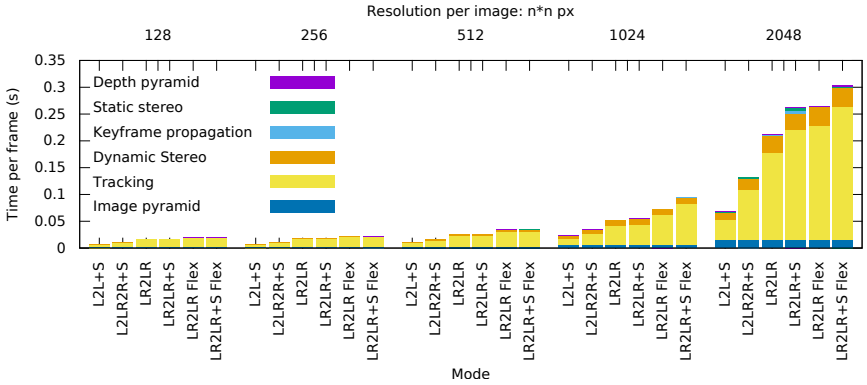

Fig. 10. Timing evaluation. The L2L+S configuration which works best for stereo rigs runs at more than 15 fps even for 4 Megapixel images. There is just small computational overhead introduced by static stereo and nonrigid tracking. Note that there is some overhead for CPU/GPU synchronization which consumes significant time for smaller resolutions. Also note that the three tasks at the top that are only performed when the keyframe is updated have a very small average per frame contribution.

small overhead for nonrigid tracking. It also shows that our GPU implementation is able to process even 4 Megapixel images in real-time.

4.4 Multi-camera rigs

To show the multi-camera capabilities of our algorithm, we evaluated it on video streams from a Point Grey Ladybug 3 which is a 6 camera, omnidirectional camera rig with slightly overlapping FoVs. We used three test scenes shown in Fig. 11. The results in Fig. 12 indicate that our tracking approach can reconstruct valid camera paths based on the data of all cameras where methods that use just one camera for tracking like [1] fail, e.g. because of few or EPL-aligned high gradient pixels or a fixed point of expansion position in the images with leads to low disparities and therefore bad depth estimations in the surrounding.

Fig. 11. Test scenes for multi-camera rig evaluation: corridor (left) exhibits few gradients, repetitive ceiling texture and glossy highlights; office (center) shows many easily trackable gradients; veranda (right) is an outdoor scene with large depth variations.

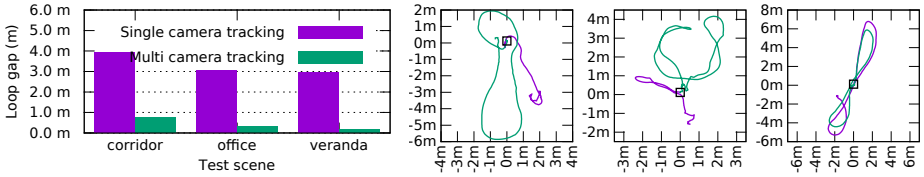

Fig. 12. Loop gap evaluation on the Point Grey Ladybug 3 which is an omnidirectional, 6 camera rig. We used static and dynamic intra-camera tracking (compare to Fig. 6, B) on a rigid rig model. Note that in contrast to stereo data with overlapping fields of view (see Fig. 7), the tracking benefits from taking all available data into account and optimizing for one consistent rig pose. The multi camera tracking reduces the error by one order of magnitude. The average processing time per frame (6 x 736x1136px) was 59ms for single camera and 120ms for multi camera tracking.

5 Conclusion

We present a direct, semi-dense visual odometry method for flexible multi-camera rigs. Key features of our method are (1) the tracking based on the photometric error that optimizes for consistent, relative rig poses as well as the rig intrinsics at once and (2) the update of the semi-dense depth with stereo information from multiple cameras in a Bayesian framework.

Our method achieves state of the art reconstruction quality for scenes recorded with completely rigid rigs. But it even supports the reconstruction with non-rigidly connected cameras. This reduces reconstruction errors dramatically, even if only small relative camera motion is present. Additionally, we show that our method is not just suitable for stereo cameras but also for other configurations like omnidirectional camera rigs with barely overlapping views.

In the future, this work could be extended to allow also for dynamic camera intrinsics as well by making the camera intrinsics a controllable parameter in the warping function (Equation 5).

Acknowledgements

This work was supported by Daimler AG, Germany. Real-world flexible stereo rig datasets were kindly provided by Dr. Senya Polikovsky, OSLab, Max Planck Institute for Intelligent Systems Tübingen.

References

1. Engel, J., Stueckler, J., Cremers, D.: Large-scale direct slam with stereo cameras. In: International Conference on Intelligent Robots and Systems (IROS). (2015)
2. Chiuso, A., Favaro, P., Jin, H., Soatto, S.: Structure from motion causally integrated over time. IEEE Trans. Pattern Anal. Mach. Intell. **24** (2002) 523–535
3. Nistér, D., Naroditsky, O., Bergen, J.: Visual odometry. (2004) 652–659
4. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: Monoslam: Real-time single camera slam. IEEE Trans. Pattern Anal. Mach. Intell. **29** (2007) 1052–1067
5. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07), Nara, Japan (2007)
6. Paz, L.M., Piniés, P., Tardós, J.D., Neira, J.: Large scale 6-dof slam with stereo-in-hand. IEEE TRANSACTIONS ON ROBOTICS **24** (2008) 946–957
7. Kerl, C., Sturm, J., Cremers, D.: Robust odometry estimation for rgb-d cameras. In: ICRA, IEEE (2013) 3748–3754
8. Meilland, M., Comport, A.I.: On unifying key-frame and voxel-based dense visual SLAM at large scales. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013. (2013) 3677–3683
9. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: European Conference on Computer Vision (ECCV). (2014)
10. Pillai, S., Ramalingam, S., Leonard, J.: High-performance and tunable stereo reconstruction. In: Robotics and Automation (ICRA), 2016 IEEE International Conference on, IEEE (2016)
11. Comport, A.I., Malis, E., Rives, P.: Accurate quadrifocal tracking for robust 3d visual odometry. In: Proceedings 2007 IEEE International Conference on Robotics and Automation. (2007) 40–45
12. Resch, B., Lensch, H.P.A., Wang, O., Pollefeys, M., Sorkine-Hornung, A.: Scalable structure from motion for densely sampled videos. In: CVPR, IEEE Computer Society (2015) 3936–3944
13. Kim, C., Zimmer, H., Pritch, Y., Sorkine-Hornung, A., Gross, M.: Scene reconstruction from high spatio-angular resolution light fields. ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH) **32** (2013) 73:1–73:12
14. Wei, J., Resch, B., Lensch, H.P.A.: Dense and occlusion-robust multi-view stereo for unstructured videos. In: 13th Conference on Computer and Robot Vision, CRV 2016, Victoria, British Columbia, June 1-3, 2016, IEEE Computer Society (2016)
15. Delaunoy, A., Pollefeys, M.: Photometric Bundle Adjustment for Dense Multi-view 3D Modeling. In: Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, IEEE (2014) 1486–1493
16. Engel, J., Sturm, J., Cremers, D.: Semi-dense visual odometry for a monocular camera. In: IEEE International Conference on Computer Vision (ICCV), Sydney, Australia (2013)
17. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quarterly Journal of Applied Mathmatics **II** (1944) 164–168
18. Crouse, D.F., Willett, P., Pattipati, K., Svensson, L.: A look at gaussian mixture reduction algorithms. In: Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on. (2011) 1–8
19. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2012)